REPLICATION AND FURTHER ANALYSIS ON PROBABILITY WEIGHTED WORD SALIENCY

Jinze Cui & Fengyi Quan
Duke University
{jinze.cui,fengyi.quan}@duke.edu

ABSTRACT

Adversarial machine learning is the study of attacks on machine learning algorithms, specifically those that involve the use of adversarial examples or inputs specifically designed to deceive models. Adversarial attacks have been well-studied in computer vision, but research in natural language processing (NLP) is more limited due to the challenges of performing attacks or defenses on text. In this paper, we focus on a recent NLP adversarial attack method called Probability Weighted Word Saliency (PWWS), which generates adversarial examples by substituting words in a given text with their synonyms. PWWS combines word saliency and classification probability to determine the word substitution order and succeeds in fooling NLP models while ensuring that the resulting adversarial examples are lexically correct without grammar mistakes or semantic shifting. The effectiveness of PWWS is demonstrated through experiments on multiple text classification datasets. The results show that our replication is able to significantly reduce the accuracy of NLP models low substitution rates. Our ablation study further investigate the source of PWWS's success.

1 INTRODUCTION

Adversary attack, a technique that tries to fool models with malformed data, is a trending topic that has been discussed heatedly in the deep learning area. A machine learning technique called adversarial machine learning tries to deceive machine learning models by giving them false information. As a result, it involves both the creation and detection of adversarial examples or inputs made specifically to trick models. These adversarial machine learning attacks have been thoroughly examined in various fields, ranging from spam detection to image classification. An adversary example that is intended to induce a machine learning model to predict incorrectly even though it would appear to be valid to a person is called an adversarial example.

In fact, deep neural networks are not resilient to these disturbances, according to recent studies that used small, imperceptible perturbations to test this hypothesis. Modern deep neural networks used for image classification were first assessed by Goodfellow et al. (2014). using modestly produced perturbations on the input images. They discovered that although there is a substantial possibility that the picture classifier was deceived, human judgment is unaffected.

However, compared with its success in Computer Vision, the related research in Natural Language Processing (NLP) is limited. Adversarial examples are inputs to machine learning models that are intentionally designed to fool the model into making incorrect predictions. In the context of NLP, adversarial examples can take the form of carefully crafted sentences or paragraphs that are semantically similar to the original input but lead the model to produce incorrect outputs. This might be because performing attacks or defenses on texts is more challenging than on image space. In the continuous space discretely labeling words in phrases related to gradient descent is more ambiguous. While the language that has been altered also has trouble being understood by human perception, little pixel alterations in photographs nonetheless result in visually appealing visuals to the human eye. However, the existence of adversary samples for NLP tasks like sentiment analysis, fake news identification, and span filtering raises worries about serious security flaws in their applications.

One of the main challenges in natural language processing (NLP) is the ability of models to generalize to unseen inputs and handle variations in language use. Adversarial examples have been

shown to be a powerful tool for evaluating the robustness and generalizability of NLP models, as they allow us to test the model's ability to correctly classify inputs that are intentionally designed to be difficult or misleading. A very important part of machine learning is to use of data sets to train models. Studying these attack methods will help us understand the principles of work on NLP and better defend against these unknown risks.

In this paper, We will focus on an ACL-2019 paper called *Generating Natural Language Adversarial Examples through Probability Weighted Word Saliency* Ren et al. (2019). It proposes a novel attacking method called Probability Weighted Word Saliency (PWWS) to generate adversarial examples on text classification tasks. The main idea of PWWS is to substitute the words in a given text with their synonyms. It combines the word saliency and the classification probability to determine the word substitution order and changes the words greedily until it can successfully fool the NLP model. The generated adversarial examples are guaranteed to be lexically correct without grammar mistakes or semantic shifting. And the attack effect of the suggested replacement word is measured using classification probability, whereas word saliency clearly illustrates how the original term influences classification. The final replacement word and its replacement sequence are determined by the change value of the classification probability weighted by word saliency.

Our goal is to reproduce PWWS and use it to generate adversarial examples for text classification tasks. We train three NLP models on text classification, including word-based CNN, LSTM and bi-directional LSTM. These models are trained and tested on three public dataset: IMDB Reviews Maas et al. (2011), AG's News Zhang et al. (2015), and Yahoo! Answers Zhang et al. (2015). We evaluate the models' performance on the original text and the generated adversarial examples. The comparison between our methods and the published results validate functionality of the replication. We also conduct further analysis to examine the source of PWWS's success.

2 Related Work

Adversarial examples are inputs to a model that have been carefully constructed to cause the model to make mistakes or misclassify the input. A brief review of some adversary works of adversary attacks is listed. There have been several approaches proposed for generating adversarial examples for natural language processing models. Let x be a sentence that we want to generate an adversarial example for, and let $f(x)$ be the model's prediction function, which maps a sentence x to a class label y. The goal of generating an adversarial example is to find a modified sentence x' that is similar to x but will cause the model to make a mistake. One common approach is to use gradient-based methods, which compute the gradient of the model's loss function with respect to the input and then modify the input in the direction that maximizes the loss. This approach has been shown to be effective in generating adversarial examples for many types of models, including neural networks (Liang et al., 2018) To generate x' is to use gradient-based methods, which compute the gradient of the model's loss function with respect to the input and then modify the input in the direction that maximizes the loss. The gradient of the loss function with respect to the input can be computed as:

$$\nabla_x L(f(x), y)$$

where $L(f(x), y)$ is the loss function that measures the difference between the model's predicted class label $f(x)$ and the true class label y. Once we have computed the loss function's gradient, we can modify the input and generate the adversarial example x'. This can be done using the following equation:

$$x' = x + \epsilon \cdot sign(\nabla_x L(f(x), y))$$

where ϵ is a small scalar value that controls the magnitude of the modification to the input. Another approach proposed for generating adversarial examples for natural language processing models is to use saliency maps. A saliency map is a visual representation of the parts of an input that are most important for the model's prediction. Saliency maps highlight the parts of an input that are most important for the model's prediction. By identifying and modifying the most important words in an input, it is possible to generate adversarial examples that are more effective at fooling the model. (Alzantot et al., 2018)

Related work in the area of adversarial machine learning for natural language processing (NLP) tasks has demonstrated successful results in generating adversarial examples, but there is still room for improvement in the percentage of modifications required to deceive the model, the success rate of the attack, maintaining lexical and syntactic correctness, and maintaining semantic similarity.

3 APPROACH

In this section, we first formulate the task of text classification attack, and then introduce the PWWS strategy. We also present additional extensions and attempts included in the project.

3.1 TEXT CLASSIFICATION ATTACK

We first clarify the notations used in our project and define the task of text classification attack.

Let $\mathbb{X} \subseteq \mathbb{R}^d$ be the feature space of the input texts, and $\mathbb{C} = \{c_1, c_2, \ldots, c_k\}$ be the label space with k possible labels in total. Giving a dataset $\mathbb{T} = \{ (\boldsymbol{x}, y) \mid \boldsymbol{x} \in \mathbb{X},\ y \in \mathbb{C}\}$, where $\boldsymbol{x}$ is a text instance and y is the correct label, the goal of a classifier is to learn a mapping $\mathcal{F} : \mathbb{X} \to \mathbb{C}$, which can classify $\boldsymbol{x}$ into the corresponding class y.

The text classification attack is a strategy to perturb a text instance $\boldsymbol{x}$ into $\boldsymbol{x}^*$ to mislead the classifier Formally, a text classification attack strategy is to learn a mapping $\mathcal{A} : \mathbb{X} \to \mathbb{X}$ such that an adversarial example

$$\boldsymbol{x}^* = \mathcal{A}(\boldsymbol{x})$$

will be wrongly labeled as $c^* \in \mathbb{C} \setminus \{y\}$. by the classifier.

In this project, we focus on the additive adversarial attack first proposed in Madry et al. (2017), where the adversarial example is an addition of a perturbation $\boldsymbol{r} \in \mathbb{X}$. In other words, $\boldsymbol{x}^*$ can only be generated by the following way

$$\boldsymbol{x}^* = \boldsymbol{x} + \boldsymbol{r}$$

To ensure the validity of the generated adversarial example, it is critical to decrease the magnitude of the perturbation $\boldsymbol{r}$ with the aim that $\boldsymbol{x}^*$ can be close to $\boldsymbol{x}$. Formally, we have the following optimization problem

$$\begin{aligned} \min_{\boldsymbol{r}} \ & ||\boldsymbol{r}||_p \\ \text{s.t.} \quad & \mathcal{F}(\boldsymbol{x}^*) \in \mathbb{C} \setminus \{y\} \\ & \boldsymbol{x}^* = \boldsymbol{x} + \boldsymbol{r} \end{aligned} \tag{1}$$

In the context of NLP, an input text instance can be regarded as a sequence of words, i.e., $\boldsymbol{x} = \langle w_1,\ w_2,\ \ldots, w_i,\ \ldots,\ w_m \rangle$, where $w_i \in \mathbb{D}$ is a single word and $\mathbb{D}$ is a dictionary. Similarly, we express the generated adversarial text as $\boldsymbol{x}^* = \langle w_1^*,\ w_2^*,\ \ldots, w_i^*,\ \ldots,\ w_m^* \rangle$. Then, the objective function (1) can be defined in the following way.

$$||\boldsymbol{r}||_p = \left(\sum_{i=1}^{m} |w_i^* - w_i^*|^p \right)^{1/p} \tag{2}$$

By minimizing the p-norm objective function (2), the perturbation $\boldsymbol{r}$ should be unnoticeable for humans. In other words, the generated adversarial examples are supposed to be lexically and grammatically correct without any significant changes in semantics.

3.2 PROBABILITY WEIGHTED WORD SALIENCY

The adversarial attack strategy proposed by Ren et al. (2019) is called Probability Weighted Word Saliency (PWWS). The general idea is to greedily replaces target words in a text instance with appropriate synonyms until it can successfully mislead the classification model. This strategy should choose a substitute synonym for each word, and decide a substitution order to minimize the perturbation. The detailed methodologies are described in the sections below.

3.2.1 SYNONYM SELECTION

The aim of synonym selection is to find the substitution word that can cause the most significant change in the classification result. It operates in the following two steps.

Building synonym sets. Following the idea in Ren et al. (2019), for any input text instance $\boldsymbol{x} = \langle w_1,\ w_2,\ \ldots,\ w_m \rangle$, we utilize WordNet Miller (1995) to create a synonym set $\mathbb{S}_i$ for each word w_i. Here, WordNet is a large lexical database for the English language, where we can find a synonyms for w_i, and include them into $\mathbb{S}_i$. Meanwhile, if w_i is a named entity, such as a person, a location, an organization and a product, $\mathbb{S}_i$ should contain the most frequent named entity with the same type in the complementary dictionary. For example, we assume the correct label of $\boldsymbol{x}$ is c_j, and $w_i \in \boldsymbol{x}$ is a person's name. Let $\mathbb{D}_{\text{Person}} \subseteq \mathbb{D}$ be a dictionary containing all person's names in the dataset, and $\mathbb{D}_{\text{Person},\ c_j} \subseteq \mathbb{D}_{\text{Person}}$ only includes person's names appearing in text instances with label c_j. Then, we choose the most frequent word in $\mathbb{D}_{\text{Person}} \setminus \mathbb{D}_{\text{Person},\ c_j}$ and add it into $\mathbb{S}_i$.

Selecting target synonym. Since each word can only be placed by at most one synonym, we must choose the one with the best attack effect among all the candidates in $\mathbb{S}_i$. Formally, we assume the decision boundaries of the classifier are specified by k discriminant functions, i.e., $g_j(\cdot)$, where $c_j \in \mathbb{C}$ and $\mathbb{C} = \{c_1, c_2, \ldots, c_k\}$. The input text $\boldsymbol{x}$ will be assigned with the correct label c_t if and only if

$$g_t(\boldsymbol{x}) \geq \max_{c_j \in \mathbb{C} \setminus \{c_t\}} g_j(\boldsymbol{x}) \tag{3}$$

Considering w_i in $\boldsymbol{x}$, we can select the target synonym w_i^* based on the following criteria

$$w_i^* = \arg\max_{w_i^s \in \mathbb{S}_i} \left(g_t(\boldsymbol{x}) - g_t(\boldsymbol{x}_i^s) \right)$$

where

$$\boldsymbol{x} = \langle w_1,\ w_2,\ \ldots, w_i,\ \ldots,\ w_m \rangle$$

and

$$\boldsymbol{x}_i^s = \langle w_1,\ w_2,\ \ldots, w_i^s,\ \ldots,\ w_m \rangle$$

By repeating the above two steps, we can choose the target synonym producing the most significant change in the classification result for each word in the input text instance.

Meanwhile, we define the attacking ability of w_i^* as

$$\Delta g_i^* = g_t(\boldsymbol{x}) - g_t(\boldsymbol{x}_i^*)$$

where

$$\boldsymbol{x}_i^* = \langle w_1,\ w_2,\ \ldots, w_i^*,\ \ldots,\ w_m \rangle$$

3.2.2 Substitution Order Calculation

With the selected target synonyms in an input text instance, we should decide the most effective order for substitution. Since each word may possess different significance, PWWS utilizes the word saliency to evaluate the word's importance on the classification result. For a given input text $\boldsymbol{x}$ with label c_t, the word saliency $S(\boldsymbol{x}, w_i)$ for any word w_i measures the degree of change in the classification result if w_i is assumed to be out-of-vocabulary (OOV) Li et al. (2016); Zhang et al. (2020). It is defined as

$$S(\boldsymbol{x}, w_i) = g_t(\boldsymbol{x}) - g_t(\boldsymbol{x}_i^{OOV})$$

where

$$\boldsymbol{x} = \langle w_1,\ w_2,\ \ldots, w_i,\ \ldots,\ w_m \rangle$$

and

$$\boldsymbol{x}_i^{OOV} = \langle w_1,\ w_2,\ \ldots, OOV,\ \ldots,\ w_m \rangle$$

Meanwhile, we define $\boldsymbol{S}_{\boldsymbol{x}}$ to be a word saliency vector containing $S(\boldsymbol{x}, w_i)$ for each w_i in the correct order. That is

$$\boldsymbol{S}_{\boldsymbol{x}} = [S(\boldsymbol{x}, w_1),\quad S(\boldsymbol{x}, w_2),\quad \cdots,\quad S(\boldsymbol{x}, w_m)]^T$$

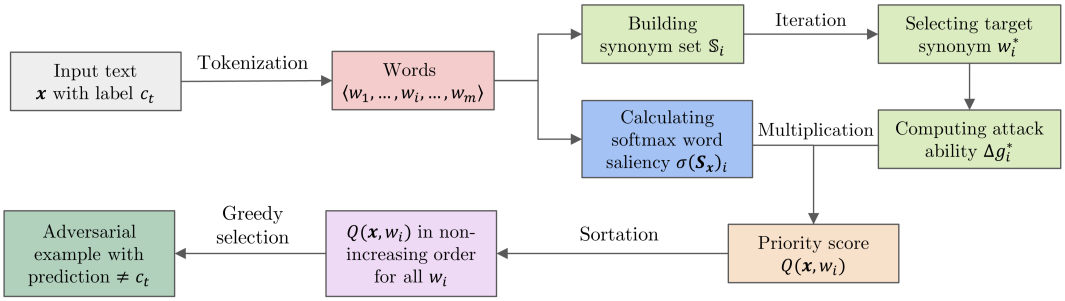

Figure 1: The entire process of PWWS

To achieve the most significant attacking effect with the fewest changes, we should consider the importance of the original word and the attacking ability of the target synonym, which are measured in $S(\boldsymbol{x}, w_i)$ and Δg_i^*, respectively. Following this idea, We define a priority score function Q to combine both of them. For each word w_i in $\boldsymbol{x}$, $Q(\boldsymbol{x}, w_i)$ is calculated as

$$Q(\boldsymbol{x}, w_i) = \sigma(\boldsymbol{S_x})_i \cdot \Delta g_i^*$$

where $\sigma(\boldsymbol{S_x})_i$ is the softmax function of $\boldsymbol{S_x}$, and it is defined as

$$\sigma(\boldsymbol{S_x})_i = \frac{\exp\left((\boldsymbol{S_x})_i\right)}{\sum_{k=1}^{m} \exp\left((\boldsymbol{S_x})_k\right)}$$

where $(\boldsymbol{S_x})_k$ represents the k-th element in $\boldsymbol{S_x}$.

After calculating $Q(\boldsymbol{x}, w_i)$ for all of the words in $\boldsymbol{x}$, we should sort them in a non-increasing order, and substitute one word with its target synonym in each iteration following the order. We should also test the classifier with this newly generated example to check whether it can deceive the classifier. This greedy algorithm will be iterated until the classifier changes its result. Formally, in contrast to the correct prediction in (3), a successful adversarial example $\boldsymbol{x}^*$ should satisfy

$$g_t(\boldsymbol{x}^*) < \max_{c_j \in \mathbb{C} \setminus \{c_t\}} g_j(\boldsymbol{x}^*)$$

The entire process to generate an adversarial example using PWWS is illustrated in Figure 1.

3.3 Extensions

Since the generated adversarial examples are supposed to remain semantically similar to the input text instances, the candidates in the synonym sets should be chosen carefully. Based on the original idea in Ren et al. (2019), if w_i is a named entity, $\mathbb{S}_i$ will include the most frequent named entity in the complementary dictionary. Although this method guarantee both of the named entities have the same type, they may be quite divergent in their meanings, and therefore, they can mislead the classifiers easily. To examine the impact of the most frequent named entity, we consider removing such words from $\mathbb{S}_i$, i.e., the synonyms are only collected through the WordNet. We also conduct an ablation study to validate our hypothesis.

4 Experiments

4.1 Datasets and Evaluation

We use the same datasets as Ren et al. (2019), including IMDB Reviews Maas et al. (2011), AG's News Zhang et al. (2015), and Yahoo! Answers Zhang et al. (2015). Table 1 lists the details of each dataset. We evaluate the effectiveness of PWWS based on the decreased amount of models' accuracies and the word substitution rate. A successful replication should reduce the accuracy to the most extent with a relatively low word substitution rate.

Dataset	#Classes	#Train samples	#Test samples	#Avg. words	Task
IMDB Review	2	25,000	25,000	325.6	Sentiment analysis
AG's News	4	120,000	7600	278.6	News categorization
Yahoo! Answers	10	1,400,000	50,000	108.4	Topic classification

Table 1: Statistics of the datasets. #Avg. words is the average number of words in each sentence.

Dataset	Model	Pub. Paper	Replication
IMDB Review	Word-based CNN	86.55%	86.62%
	Bi-dir. LSTM	84.86%	84.59%
AG's News	Word-based CNN	90.56%	90.74%
Yahoo! Answers	Word-based CNN	96.01%	96.31%
	LSTM	92.00%	90.15%

Table 2: Classification accuracies in the published paper and our replication. Bi-dir. LSTM refers to the bi-directional LSTM model, and Pub. paper refers to the results in Ren et al. (2019).

4.2 Models and Classification Results

Following the description in Ren et al. (2019), we developed three models for text classification.

- **Word-based CNN** This model includes a 50-dimensional word embedding layer, a 1D-convolutional layer consisting of 250 filters with 3 kernel size, a max-pooling layer, and two fully connected layers with a ReLU function.
- **LSTM** This model consists of an embedding layer performing 100-dimensional word embeddings, an LSTM layer with hidden size of 128, and one fully connected layer.
- **Bi-directional LSTM** This model is composed by an embedding layer with 128-dimensional word embeddings, a Bi-directional LSTM layer with hidden size of 64, and one fully connected layer.

All of these models choose Adam as the optimizer and cross entropy as the loss function. Similar to the deployment in the original paper, we use LSTM on Yahoo! Answers dataset, Bidirectional LSTM on IMDB Review, and Word-based CNN on all three datasets. The performance comparison between our models and the results published in Ren et al. (2019) is summarized in Table 2.

4.3 PWWS Attack Results

We use PWWS to generate 2,000 adversarial examples for each model on the corresponding dataset. Its performance is evaluated by the model prediction accuracy and the substitution rate, which is the number of substituted words divided by the total number of words. The comparison between the published results and our replication is summarized in Table 3. We also present two adversarial examples in Table 4 to illustrate the changes in model predictions and text instances.

5 Analysis

In this section, we analyze the results of our replication and conduct an ablation study to examine a possible success factor of PWWS.

5.1 Discussion on the Replication Results

Results in Section 4.2 and 4.3 indicate that our project is a valid replication of the classification models and PWWS proposed in Ren et al. (2019). As shown Table 2 and 3, although there exist gaps

Dataset	Model	Accuracy		Substitution Rate	
		Pub. paper	Replication	Pub. paper	Replication
IMDB Review	Word-based CNN	5.50%	6.15%	3.81%	4.76%
	Bi-dir. LSTM	2.00%	2.35%	3.38%	4.21%
AG's News	Word-based CNN	56.72%	55.35%	16.76%	15.49%
Yahoo! Answers	Word-based CNN	57.71%	60.15%	25.43%	27.18%
	LSTM	53.00%	56.55%	35.10%	36.73%

Table 3: Classification accuracies on the adversarial examples generated by PWWS and the word substitution rates. Bi-dir. LSTM refers to the bi-directional LSTM model, and Pub. paper refers to the results in Ren et al. (2019).

Dataset	Model	Prediction Shift	Text Changes
IMDB Review	Word-based CNN	Positive → Negative	It has been years since I have been privileged selected to see this movie . . . but still has many after affects of being born by an alcoholic drunk mother. . . . this has truly made monumental massive changes in their behaviors to start out a new life. . .
Yahoo! Answers	LSTM	Business and Finance → Social Science	Are cigarette makers being unfairly taxed . . . Tobacco advertising arguments for and against drug prohibition lung cancer crab . . . John Wayne Life insurance policy at Durham . . . The one with the colorectal cancer crab and . . .

Table 4: Adversarial examples generated by PWWS. Column 3 refers to the shift of the model predictions from the original text to the adversarial text. Column 4 shows the changes in the input text. Words in orange are original while words in blue are the substitutions by PWWS.

in prediction accuracies and substitution rates between our replication and the published results, the values are not significant. Since we do not have access to the detailed training configurations and the exact train-test splits, this might be a possible reason for the slightly different performance.

5.2 Ablation Study on Named Entities

As mentioned in Section 3.3, we suspect that adding the most frequent name entity in the synonym set may violate the semantic constraint, and therefore, it can be a major source of PWWS's success. To further investigate this idea, we conduct ablation study by building synonym sets only through the WordNet. The results are shown in Table 5.

We observe that after removing the most frequent named entity, the model prediction accuracies significantly increase. It indicates that the most frequent named entities is of critical importance to PWWS. We may conduct further research to examine whether these named entities can lead to the semantic divergence and design a new method to deal with them more carefully.

6 Conclusion

In this project, we replicate the Probability Weighted Word Saliency proposed in Ren et al. (2019). We evaluate it using Word-based CNN, LSTM and bi-directional LSTM on IMDB Reviews, AG's News and Yahoo! Answers. The significantly decreased classification results and low substitution rates validate the effectiveness of our replication. It also achieves competitive performance com-

Dataset	Model	WordNet + Most Fret. NE	WordNet Only
IMDB Review	Word-based CNN	6.15%	8.80%
	Bi-dir. LSTM	2.35%	5.35%
AG's News	Word-based CNN	55.35%	63.15%
Yahoo! Answers	Word-based CNN	60.15%	67.95%
	LSTM	56.55%	66.40%

Table 5: The ablation study on the named entities in the synonym set. Results in this table are classification accuracies on the adversarial examples. Column 3 refers to synonym sets including both the WordNet synonyms and the most frequent named entity in the complementary dictionary, while column 4 refers to synonym sets built through the WordNet only.

pared with the published results in the original paper. Meanwhile, we conduct an ablation study to validate the crucial importance of the most frequent named entities in the synonym sets.

Author Contributions

- Jinze Cui develops PWWS and evaluate its performance. He writes Section 3 Approach, 4.2 Models and Classification Results, 4.3 PWWS Attack Results, 5 Analysis, and 6 Conclusion.
- Fengyi Quan processes datasets and trains three classification models. He writes Abstract, Section 1 Introduction, 2 Related Work, and 4.1 Datasets and Evaluation.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2890–2896, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1316. URL https://aclanthology.org/D18-1316.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2014.

Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding neural networks through representation erasure. *CoRR*, abs/1612.08220, 2016. URL http://arxiv.org/abs/1612.08220.

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. Deep text classification can be fooled. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, pp. 4208–4215. AAAI Press, 2018. ISBN 9780999241127.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1015.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11): 39–41, 1995.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1085–1097, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1103. URL https://aclanthology.org/P19-1103.

Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(3):1–41, 2020.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NIPS*, 2015.